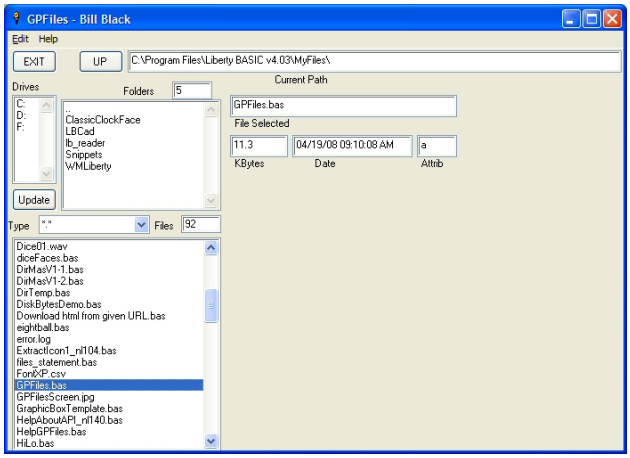
GPFiles - A General Purpose Interface Using the Files Command by - BillBlack

In most programs, the LB command 'filedialog' works well to open and save files at the directory location the user chooses. However, for some applications, it would be better to have the directory and file structure continually displayed so the user can navigate to a location, perform the desired action and move quickly to another location without waiting to execute 'filedialog' again. The LB command 'files' is well suited to this type of interface between the application and the computer's directory structure.
I actually wrote two programs that utilized 'files' this way- one to search for and view text based files (bas, txt, ini, bat, etc) or graphic files (bmp, jpeg, etc) and the other was to preview and print bas type files. It then occurred to me that I should back up and produce a general purpose 'files' program that could be easily expanded with a few additional controls for a specific application. This general purpose program I called GPFiles and is the subject of this article on the LBPE.
Here are the requirements I wanted and, I think, obtained for GPFiles:
1. A Windows Explorer type ability to navigate the directory structure.
2. A continuous display of the current path.
3. A display and count of the folders in the current path.
4. A display and count of the files in the current path.
5. A display of the current drives.
6. An ability to update the drives if another drive is added or removed.
7. A filetype mask to limit the files displayed to that type.
8. A file selected box to display size, date and attributes about that file.
9. A large area for future application controls.
10. Selection/navigation for listboxes would be double-click.

Here is a screen shot of the interface:



GPFiles Interface Screen

Most of the program is very standard LB use of listboxes and writing to text boxes and a small help screen is included to explain use. There are two parts that should be explained.
1. The [bUpDrvClk] branch updates the Drives array drvs$. Credit for this goes to Alyce Watson who covered it nicely in newsletter 142 in the article "API Corner - Drive Strings." It involves the use of a dll in kernel32 named 'GetLogicalDriveStringsA." She covers this very well in that article so check there for more information. Anyway, it retrieves a current list of drives installed on the computer so it will add or

delete drives, like a camera or a USB thumbdrive, which has been plugged in or removed while the program is running.
2. The [getDirsFiles] subroutine is at the heart of this program and utilizes the 'files' command to retrieve all of the folders (subdirectories) and files in the selected path. This command is covered well in the LB help files. The only setup needed to use 'files' is to declare a two dimensional string array. This array is passed to the 'files' function and is redimensioned upon return to the program and contains all the file and folder information in the path requested. Here is a table showing how the info is returned in the array:

	column 0	column 1	column 2	column 3
row 0	#files(f)	#dirs(d)		
row 1	fnam.1	fsiz.1	fdat.1	fatt.1
row2	fnam.2	fsiz.2	fdat.2	fatt.2
row ..	..	..	..	..
row n=#f	fnam.n	fsiz.n	fdat.n	fatt.n
row n+1=#f+1	dnam.1			
row n+2	dnam.2			
row ..	..			
row m=#f+#d	dnam.m			

I did learn something about 'files' that isn't shown in the help file or mentioned elsewhere that I could find. If arrayNames() is the array passed, then, on return, arrayNames(i+1,0) contains a filename, arrayNames(i+1,1) contains a filesize, arrayNames(i+1,2) contains a file date and, surprise, arrayNames(i+1,3) contains the attributes of the file such as 'a' for archive, 'r' for read only, etc.
Here is a brief review of how [getDirsFiles] works:

```
[getDirsFiles]
'Code added for multiple file types
tmask$=mask$
if wc(mask$,";") > 1 then mask$="*.*"
'end added code
dim arrayName$(10, 10)
files curDir$, mask$, arrayName$()
```

After writing this program, I discovered that the files command would not use multiple file types in the mask unlike filedialog. The first two code lines test for this by counting the words in the mask with the wc function and, if there are more than one, it sets the mask equal to *.* for all files. The files command is called with the current directory, mask and array for results.

```
qtyFiles = val(arrayName$(0,0))
qtySubDirs = val(arrayName$(0,1))
if qtyFiles=0 then files$(0)="": goto [subdir]
if qtyFiles < fMax-1 then
for i=0 to qtyFiles-1
files$(i) = arrayName$(i+1,0)
fsize$(i) = arrayName$(i+1,1)
fdate$(i) = arrayName$(i+1,2)
fattr$(i) = arrayName$(i+1,3)
next
else
t$="Too Many Files: "+str$(qtyFiles)
notice t$
qtyFiles=fMax-1
goto [subdir1]
end if
```

Next, the quantity of files and subdirectories is retrieved. If the quantity of files is zero, then the first entry in the files$ area is set to null and we skip on to check the subdirectory entry. Otherwise all of the info for the file entries is copied into the files$, fsize$, fdate$ and fattr$ arrays. If the number of files is greater than fMax-1 (fmax is currently set at 2500 but can be easily increased) then a notice is displayed, the quantity of files is reset to fMax-1 and we skip past the nulling routine as it is not needed then.

```
[subdir]
for i=qtyFiles to fMax-1 'null remaining files
files$(i)=""
fsize$(i) = ""
fdate$(i) = ""
fattr$(i) = ""
next
```

This section merely nulls out any remaining items in the files arrays so previous larger entries won't show in the listbox.

```
[subdir1]
folds$(0) = ".."
if qtySubDirs=0 then [subdir2]
if qtySubDirs < fMax-1 then
for i=qtyFiles+1 to qtyFiles + qtySubDirs
folds$(i-qtyFiles) = arrayName$(i,1)
next
else
t$="Too Many Directorys: "+str$(qtySubDirs)
notice t$
qtySubDirs=fMax-1
end if
[subdir2]
for i=qtySubDirs+1 to fMax-1
folds$(i) = ""
next
```

This subdirectory portion essentially follows the pattern of the files section. However, the first entry is set to ".." to use later to navigate up the path. (Old users, like me, of MSDOS will recognize this from such commands as 'cd ..' meaning move up the path one level.) As before, tests are made for no entries and more entries than fMax-1 and appropriate action is taken. And again, the remaining entries in the fold$ array are nulled.

```
'Code added for multiple file types
if wc(tmask$,";") = 1 then [skipAdded]
nfiles=0
mask$=tmask$
for i=0 to qtyFiles-1
a$=lower$(re2$(files$(i),"."))
for j=1 to wc(tmask$,";")
b$=re2$(word$(mask$,j,";"),".")
if a$=b$ then
files$(nfiles)=files$(i)
fsize$(nfiles) = fsize$(i)
fdate$(nfiles) = fdate$(i)
fattr$(nfiles) = fattr$(i)
nfiles=nfiles+1
end if
next
next
for i=nfiles to fMax-1 'null remaining files
files$(i)=""
fsize$(i) = ""
fdate$(i) = ""
fattr$(i) = ""
```

```
next
qtyFiles=nfiles
[skipAdded]
'end added code
return
```

The last section is again for the added code to handle masks with more that one filetype. Since our first test for multiple filetypes changed the mask to all types, we loop thru the file arrays for names, size, date and attributes, checking for the filetypes in the mask. If it is a match, then we copy the desired data and increment the matched counter nfiles. If it doesn't match, then we skip the copy portion and don't increment the counter. All of this simply repacks the data for the desired file types into the same arrays and we finish by nulling out the rest of the arrays. Finally, the subroutine returns to the place in the program that called it.
A note on execution speed:
I was initially concerned that displaying all of these file and folders would be unacceptably slow but that is not the case. For example, the largest file list on my computer (Dell Inspiron, 1.4 gHz, Intel Celeron and Windows XP Home Ed SP2) is over 2400 files in C:\WINDOWS\SYSTEM32\. This update and display takes much less than a second to accomplish. Most smaller directory and file structures of a few hundred display immediately after double clicking the folder desired or the UP button. If you find you have longer structures, just change the fMax variable. Also, if you want different filetype masks, just change the data statements near the end of the program and adjust the size of the msk$() array if needed.
Finally, I hope to submit a followup article for an application which I have written using GPFiles to respond to a challenge issued by Alyce Watson in newsletter 104 regarding icons. Here is her remark:
"To simplify our demo, we are passing an index of 0 to extract the first icon in the disk file. We could instead, use an index of -1 to obtain the number of icons in the file, then allow the user to choose the desired icon in a preview window. Perhaps some intrepid soul will modify the demo to do just that, and share it with the group?"

- BillBlack - (intrepid soul in training)

Liberty Basic User Group Name: BillBlack

Here is the entire program:

```
'Form created with the help of Freeform 3 v03-27-03

    nomainwin
    WindowWidth = 700
    WindowHeight = 505
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)
```

```
    fMax=2500        'File quan. limit
    dim files$(fMax) 'Current file names
    dim fsize$(fMax) 'Current file sizes
    dim fdate$(fMax) 'Current file dates
    dim fattr$(fMax) 'Current file attributes
    dim folds$(fMax) 'Current folders
    dim drvs$(25)    'Current drives
    dim msk$(25)     'File type array
    curPath$=""     'Current Path eg. C:\dir1\dir2\file.ext
    curFile$=""     'Current File eg. file.ext
    curDir$=""      'Current Directory eg. C:\dir1\dir2\
    foldSel$=""     'Currently selected folder
    fileSel$=""     'Currently selected file
    drvSel$=""      'Currently selected drive
    qtyFiles=0      'file count in last FILES function
    qtySubDirs=0    'sub-directory ount in last FILES function
    index = 0       '***** load drives array
    gosub [loadhelp] 'Set up simple help array
    index = 0       '***** load drives array
    while word$(Drives$, index + 1) <> ""
        drvs$(index) = upper$(word$(Drives$, index + 1))
        index = index + 1
    wend
    gosub [getmsk]
    mask$ = msk$(0)   'file selection mask
    curPath$ = DefaultDir$+"\"+"text.txt" ' ***** load current dir and
file
    curFile$= word$(curPath$,wordcount(curPath$,"\"), "\")
    curDir$ = word$(curPath$,1,curFile$)
    gosub [getDirsFiles]    'load current subdir's and files

    '-----Begin GUI objects code
    listbox #1.drvs, drvs$(, [drvsClick],    5,  52,  50, 100
    listbox #1.folds, folds$(, [foldsClick],   60,  57, 180, 125
    listbox #1.files, files$(, [filesClick],    5, 212, 235, 240
    textbox #1.tbCurPath, 135,   2, 555,  25
    textbox #1.tbCurFile, 250,  52, 255,  25
    button #1.bUpDrv,"Update",[bUpDrvClk], UL,   5, 157,  50,  25
    statictext #1.st16, "Current Path", 365,  32,  72,  20
    statictext #1.st17, "Drives",   5,  32,  40,  20
    statictext #1.st18, "Folders", 130,  37,  55,  20
    statictext #1.st19, "Files", 165, 192,  30,  15
    statictext #1.st21, "File Selected", 255,  77,  80,  18
    textbox #1.tbFolds, 185,  37,  45,  20
    textbox #1.tbFiles, 195, 187,  45,  20
    combobox #1.cbMsk, msk$(, [cbMskDoubleClick],   35, 187, 125, 100
```

```
    statictext #1.st33, "Type",   0, 192,  30,  20
    textbox #1.tbCurSiz, 250,  97,  65,  25
    textbox #1.tbCurDat, 320,  97, 135,  25
    textbox #1.tbCurAtt, 460,  97,  45,  25
    statictext #1.st34, "KBytes", 255, 122,  50,  20
    statictext #1.st35, "Date", 345, 122,  40,  20
    statictext #1.st36, "Attrib", 465, 122,  40,  20
    button #1.bUp,"UP ",[bUpClick], UL,  80,   2,  50,  25
    button #1.bUp,"EXIT",[bmpExitClick], UL,   5,   2,  50,  25
    menu #1, "Edit"
    menu #1, "Help",_
             "Help" , [help],_
             "About", [about]
    open "GPFiles - Bill Black" for window as #1
    print #1, "trapclose [quit.1]"
    print #1, "font ms_sans_serif 6"
    #1.cbMsk, "selectindex 1"

[UpDateCurPath]
    #1.tbCurPath, curDir$
    #1.tbFolds, str$(qtySubDirs)
    #1.tbFiles, str$(qtyFiles)
    wait

[help]
      oldx=UpperLeftX
      oldy=UpperLeftY
      oldww=WindowWidth
      oldwh=WindowHeight
      UpperLeftX = 32
      UpperLeftY = 52
      WindowWidth = 700
      WindowHeight = 500
    open "Help" for text as #2
    #2, "!trapclose [quithelp]"
    #2, help$
    wait

[quithelp]
      UpperLeftX = oldx
      UpperLeftY = oldy
      WindowWidth = oldww
      WindowHeight = oldwh

    close #2
    wait
```

```
[about]
    'Thanks to Alyce Watson
    szApp$="GPFiles # GPFiles - A General Purpose Files template!"
    cr$ = chr$(13)  'carriage return
    szOtherStuff$ = cr$ + "Created by Bill Black" + cr$
    hIcon=0
    hWnd=0

calldll #shell32, "ShellAboutA",_
    hWnd as ulong,_
    szApp$ as ptr,_
    szOtherStuff$ as ptr,_
    hIcon as ulong,_
    ret as long
    wait

[drvsClick]   'change drive
    #1.drvs, "selection? drvSel$"
    if drvSel$="" then notice "select a drive": goto [enddrvsClick]
    mask$=msk$(0)
    #1.cbMsk, "selectindex 1"
    curDir$=drvSel$ + "\"
    gosub [getDirsFiles]
    gosub [updateAll]
  [enddrvsClick]
    wait

[foldsClick]   'update directories
    #1.folds, "selection? foldSel$"
    if foldSel$ = ".." then
        goto [bUpClick]
    else
        'mask$=msk$(0)             'uncomment line to restore auto re
vert to *.* filetype
        '#1.cbMsk, "selectindex 1"  'uncomment line to restore auto re
vert to *.* filetype
        curDir$=curDir$+foldSel$+"\"
        gosub [getDirsFiles]
        gosub [updateAll]
    end if
[endfoldsClick] wait

[bUpClick]   'Move up path
    n=wordcount(curDir$,"\")
    if n=1 then [skip]
```

```
    'mask$=msk$(0)                  'uncomment line to restore auto re
vert to *.* filetype
    '#1.cbMsk, "selectindex 1"      'uncomment line to restore auto re
vert to *.* filetype
    t$=""
    for i=1 to n-1
    t$=t$+word$(curDir$,i,"\")+"\"
    next
    curDir$=t$
  [skip]
    gosub [getDirsFiles]
    gosub [updateAll]
    wait

[filesClick]   'Select file with double click
    #1.files, "selection? fileSel$"
    #1.files, "selectionindex? ix"
    #1.tbCurFile, fileSel$
    #1.tbCurSiz, str$(int(val(fsize$(ix-1))/100 + .5)/10)
    #1.tbCurDat, fdate$(ix-1)
    #1.tbCurAtt, fattr$(ix-1)
    wait

[bUpDrvClk]   'Update drives in case another inserted/removed
    for i=0 to 24: drvs$(i)="": next
    dS$ = space$(2)
    LB = len(dS$)
    calldll #kernel32, "GetLogicalDriveStringsA",_
    LB as ulong,_   'length of buffer
    dS$ as ptr,_    'buffer
    nByte as ulong  'size of buffer needed
    dS$ = space$(nByte+1)   'resize buffer
    LB = len(dS$)
    calldll #kernel32, "GetLogicalDriveStringsA",_  'call again
    LB as ulong,_   'length of buffer
    dS$ as ptr,_    'buffer
    nByte as ulong  'length of string returned
    index=0 'reload drvs$
    while word$(dS$, index + 1) <> ""
        drvs$(index) = left$(upper$(word$(dS$, index + 1)),2)
        index = index + 1
    wend
    #1.drvs, "reload"
    wait

[cbMskDoubleClick]   'change filetype mask
```

```
    'Insert your own code here
    #1.cbMsk, "contents? mask$"
    gosub [getDirsFiles]
    #1.folds, "reload"
    #1.files, "reload"
    #1.tbCurPath, curDir$
    #1.tbFolds, str$(qtySubDirs)
    #1.tbFiles, str$(qtyFiles)
    fileSel$=""
    #1.tbCurFile,fileSel$
    wait

[bmpExitClick]
[quit.1]
    close #1
    end

'Functions and Subroutines
function wordcount(cnt$, delimiter$)
    cnt = 0
    work$ = ""
    while len(work$) < len(cnt$)
        cnt = cnt + 1
        work$ = work$ + word$(cnt$,cnt,delimiter$) + delimiter$
    wend
    wordcount = cnt
 end function

[getDirsFiles]
    'Code added for multiple file types
    tmask$=mask$
    if wc(mask$,";") > 1 then  mask$="*.*"
    'end added code
    dim arrayName$(10, 10)
    files curDir$, mask$, arrayName$()
    qtyFiles = val(arrayName$(0,0))
    qtySubDirs = val(arrayName$(0,1))
    if qtyFiles=0 then files$(0)="": goto [subdir]
    if qtyFiles < fMax-1 then
        for i=0 to qtyFiles-1
            files$(i) = arrayName$(i+1,0)
            fsize$(i) = arrayName$(i+1,1)
            fdate$(i) = arrayName$(i+1,2)
            fattr$(i) = arrayName$(i+1,3)
        next
    else
```

```
        t$="Too Many Files: "+str$(qtyFiles)
        notice t$
        qtyFiles=fMax-1
        goto [subdir1]
    end if
[subdir]
    for i=qtyFiles to fMax-1  'null remaining files
        files$(i)=""
        fsize$(i) = ""
        fdate$(i) = ""
        fattr$(i) = ""
    next
[subdir1]
    folds$(0) = ".."
    if qtySubDirs=0 then [subdir2]
    if qtySubDirs < fMax-1 then
        for i=qtyFiles+1 to qtyFiles + qtySubDirs
            folds$(i-qtyFiles) = arrayName$(i,1)
        next
    else
        t$="Too Many Directorys: "+str$(qtySubDirs)
        notice t$
        qtySubDirs=fMax-1
    end if
[subdir2]
    for i=qtySubDirs+1 to fMax-1
        folds$(i) = ""
    next

    'Code added for multiple file types
    if wc(tmask$,";") = 1 then [skipAdded]
    nfiles=0
    mask$=tmask$
    for i=0 to qtyFiles-1
        a$=lower$(re2$(files$(i),"."))
        for j=1 to wc(tmask$,";")
            b$=re2$(word$(mask$,j,";"),".")
            if a$=b$ then
            files$(nfiles)=files$(i)
            fsize$(nfiles) = fsize$(i)
            fdate$(nfiles) = fdate$(i)
            fattr$(nfiles) = fattr$(i)
            nfiles=nfiles+1
            end if
        next
    next
```

```
    for i=nfiles to fMax-1    'null remaining files
        files$(i)=""
        fsize$(i) = ""
        fdate$(i) = ""
        fattr$(i) = ""
    next
    qtyFiles=nfiles

    [skipAdded]
    'end added code
 return

[getmsk]
data "*.*","*.ico;*.exe;*.dll;*.icl","*.txt","*.ini","*.bas"
data "*.bmp","*.htm*","*.doc","*.ffa"
data "*.jpg","*.pdf","*.exe","*.dll","*.ico","*.icl","end"

t$="":i=0
while t$ <> "end"
    read t$
    msk$(i)=t$
    i=i+1
wend
i=i-1: msk$(i)=""
return

[updateAll]
    #1.folds, "reload"
    #1.files, "reload"
    #1.tbCurPath, curDir$
    #1.tbFolds, str$(qtySubDirs)
    #1.tbFiles, str$(qtyFiles)
    fileSel$=""
    #1.tbCurFile,fileSel$
return

function wc(st$,msk$)
t$="x": i=0
while t$ <> ""
    i=i+1
    t$=word$(st$,i,msk$)
wend
wc=i-1
end function

function re2$(st$,msk$)
```

```
    re2$=word$(st$,2,msk$)
end function

[loadhelp]
    cr$=chr$(13)
    help$=" "+cr$
    help$=help$+cr$
    help$=help$+"                 Help for GPFiles"+cr$
    help$=help$+cr$
    help$=help$+"  GPFiles is intended as a General Purpose Files and
Directory"+cr$
    help$=help$+"  interface for other applications and works much lik
e Windows"+cr$
    help$=help$+"  Explorer."+cr$
    help$=help$+cr$
    help$=help$+"  -The current directory path is always displayed at
the top and the"+cr$
    help$=help$+"    names and quantity of folders and files are displ
ayed for that path."+cr$
    help$=help$+"  -Navigate up the directory path by clicking the UP
button or double-"+cr$
    help$=help$+"    clicking the double dot (..) entry listed at the
top of folders."+cr$
    help$=help$+"  -Navigate down the directory path by double-
clicking on one of the"+cr$
    help$=help$+"    directory names in the folders list."+cr$
    help$=help$+"  -Select a file by double-
clicking on one of the names in the files list."+cr$
    help$=help$+"    The file name, size, date and attributes will be
displayed."+cr$
    help$=help$+"  -Select a drive by double-
clicking on one of the drive letters."+cr$
    help$=help$+"    If you click on the Update button under the drive
 list, it will"+cr$
    help$=help$+"    update the drive list even if you have added or r
emoved a drive"+cr$
    help$=help$+"    while the program is running."+cr$
    help$=help$+"  -The empty area of the window is intended for contr
ols for applications"+cr$
    help$=help$+"    such as a file or graphics viewer or icon preview
er/saver."
return
```